A Linux Power Management Architecture

Matthew Locke

November 2007

Embedded Alley

Agenda

- Introduction
- Features in Kernel and User space
- Pulling it together
- Future work

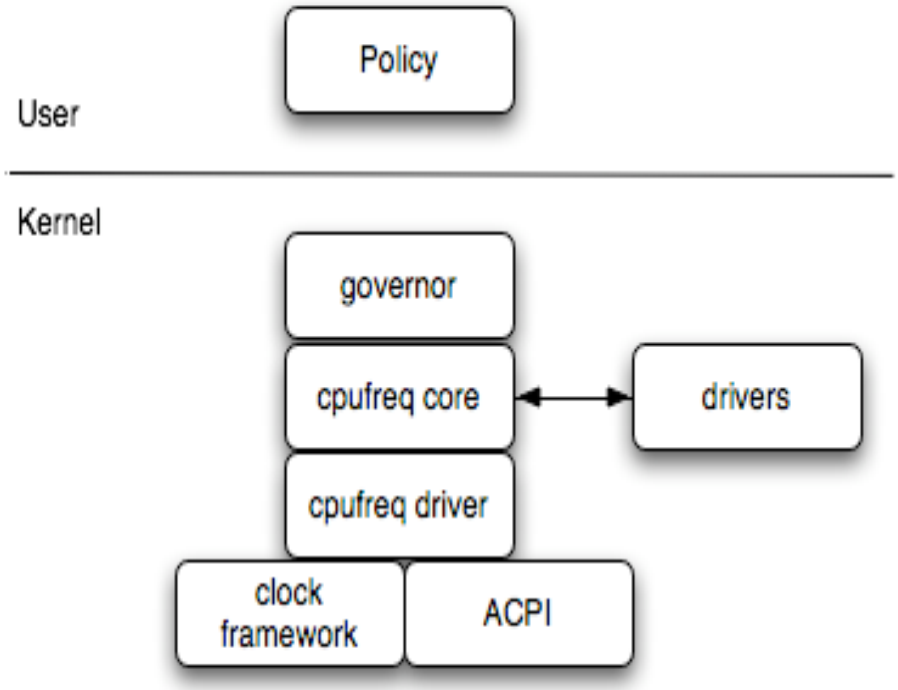
Introduction

- Typical mainline Linux power management features:
  - Per platform idle loop allows platforms to place processor in a low power state
  - Suspend-to-RAM - memory in auto refresh, CPU in a low power state, drivers in a low power state
  - Cpu frequency scaling
- Of course, main target of these features is the x86 laptop
- Power management for embedded mobile devices has been custom development per device and different for every SoC
- Over the last year, the mailing list and development activity has increased dramatically
- 2nd Linux PM Summit was held this year

System Suspend/Resume

- Recently the system suspend/resume code was redesigned
- Better support for platform specific behavior
- Better support for hibernation (suspend-to-disk)
- Pm_ops were reworked to ensure system follows correct steps to prepare for a suspend.
- Suspend-to-disk code is renamed to hibernation with a clear distinction from suspend. Also, being reworked to ensure memory snapshot, userspace and drivers are all handled correctly for stable behavior.
- **Expect longer life and more stable behavior on your laptop in the next couple releases!**

Cpufreq



- The cpufreq stack manages the runtime power management for the CPU.
- Some ACPI platforms trigger voltage changes based on CPU frequency changes.
- The embedded platforms do not have a mechanism to change voltage
- The “on-demand” governor changes cpu frequency based on load.
- Cpufreq is connected to the clock framework only on OMAP1 and OMAP2

Dynamic Tick / clockevent

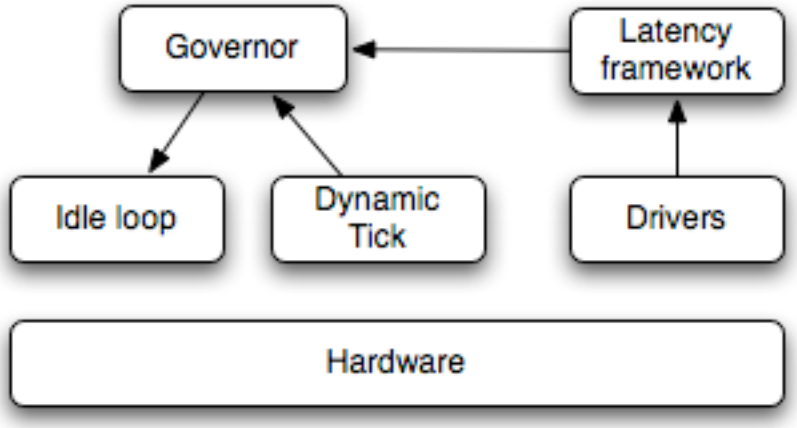
- The whole time subsystem was redesigned to eliminate the periodic timer which is very good for power management.
- Clockevent is the bottom layer with High Resolution Timers and time subsystem on top.
- Now there is an optimized platform independent way to find the next event
- Platform idle loop uses this standard API to find the next event and decide the course of action

Latency Framework

- Tracks minimal latency the system (including all drivers) can tolerate in order not to break.
- API includes
  - Register driver/subsystem with framework
  - Set a latency constraint
  - Subscribe to notification for latency changes
  - Get system wide latency constraint
- Only a few drivers use the latency framework. It will take a while for driver maintainers to update.
- An example is an audio driver that knows it will get an interrupt when the hardware has 200 usec of samples left in the DMA buffer; in that case the driver can set a latency constraint of, say, 150 usec.
- Reworked into pm_qos patches

CPUidle Framework

- Framework for selecting optimal CPU power state in the idle loop.

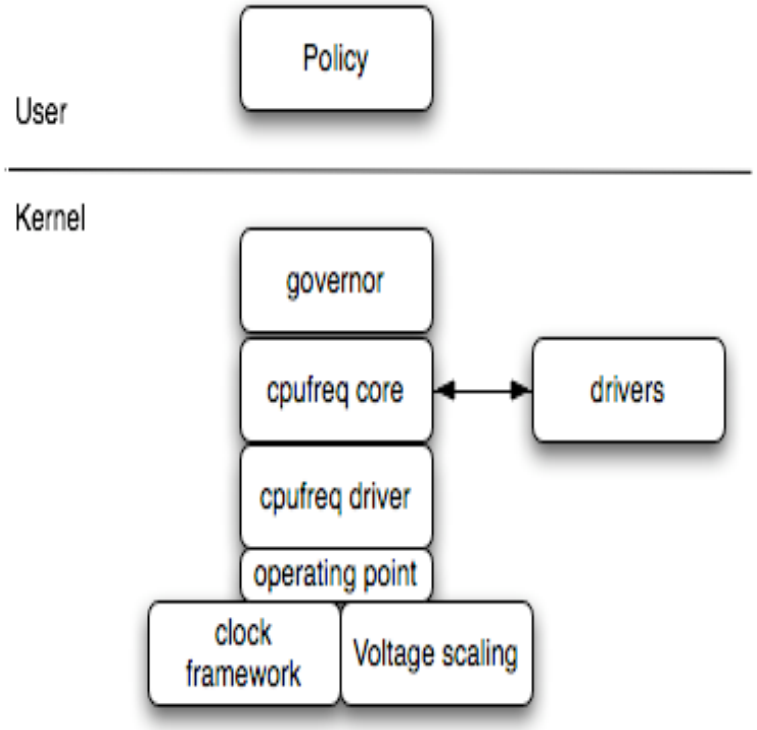


- CPU power states are defined by descriptors
- Builds on dynamic tick and latency framework

CPUidle Framework

- Two different governors are provided:
  - Menu - analyzes the latency from the latency framework, latency from the descriptor and the next clockevent from dynamic tick to determine the lowest cpu state possible
  - Ladder - uses activity metrics to step the CPU power state into the right mode. Mostly applicable to ACPI platform

Operating Points



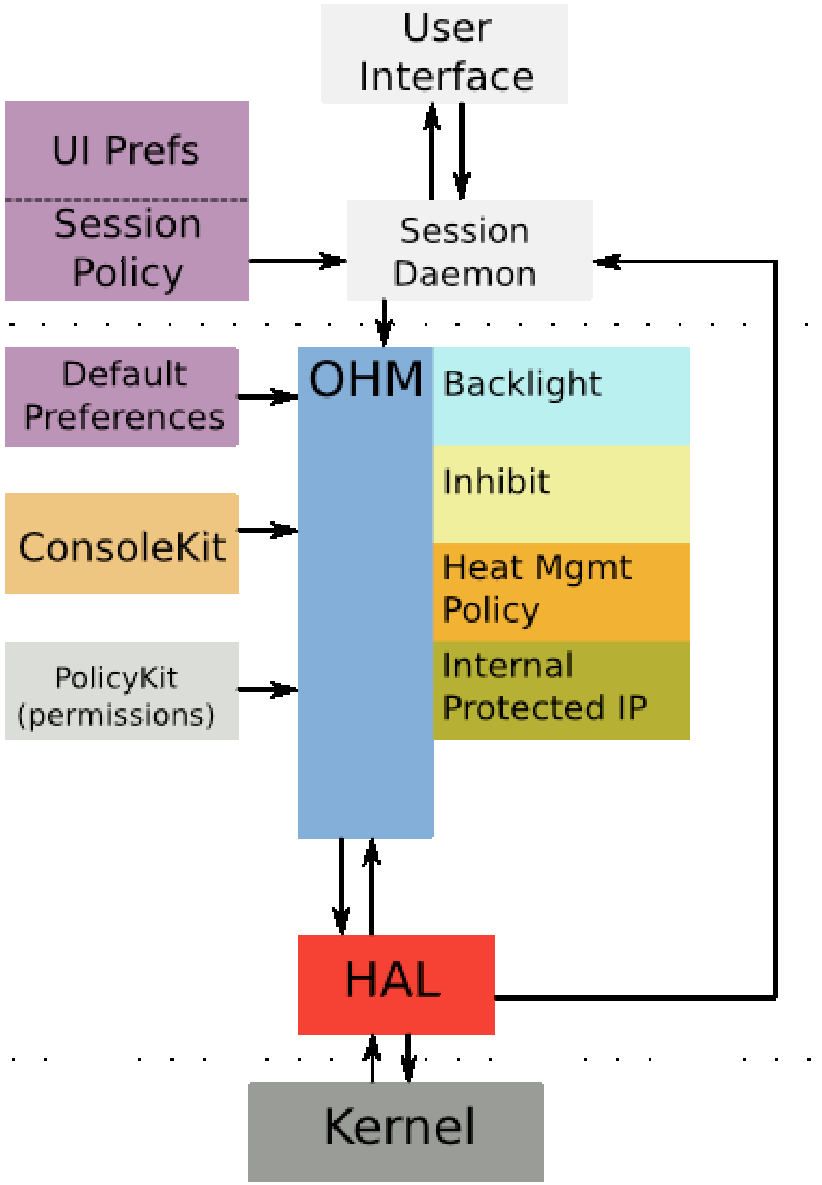
Current thinking is to incorporate voltage scaling into the lower layer of cpufreq

- Operating Point is a group of power parameters set to specific values.
- Parameters are CPU clock and voltage but may include other parameters such as bus clocks.
- Run time power consumption can be reduced by lowering voltage and frequency
- Several attempts were made over the last year to mainline a operating point implementation but were rejected
- On x86 operating points are hidden in ACPI so the challenge is to get a solution that works for other platforms without affecting x86.

Device runtime power management

- Userspace control over driver power management is being deprecated
  - /sys/…/power/state file is removed
- Drivers are expected to manage their device power management state during runtime to minimize power consumption.
- Philosophy is that drivers know best when and how to put a device into a low power state during runtime
- USB stack is leading this effort. USB stack has a autosuspend/resume feature for the host and devices. It watches for inactivity and turns stuff off.

OHM - Open Hardware Manager



- OHM is a small open source systems daemon which sits above HAL and abstracts out common hardware management tasks such as system wide inhibit action control
- The main use cases described are taking action based on subsystem inactivity
- Used by OLPC
- From OHM website

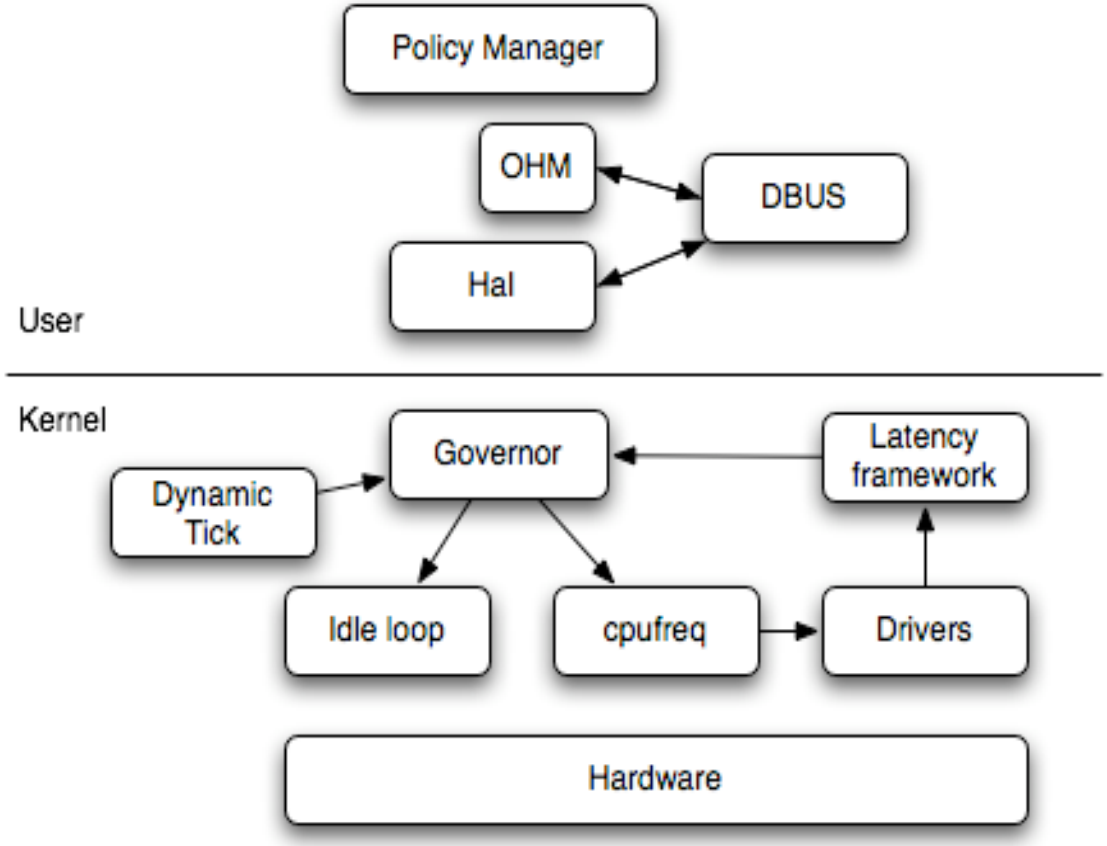
Embedded Alley

OLPC, maemo, and Moblin

- According to material presented at the PM summit '07, Nokia's stack looks similar to picture on previous slide
- Moblin is a Intel sponsored/funded project for a internet tablet application stack. A Policy manager is a part of the stack and is currently being designed.
- OLPC is working a policy manager based on OHM

Pulling it all together…

- A pm architecture using the latest components enables some very aggressive policies to reduce power consumption.



Embedded Alley

Pulling it all together…

- Automatically setting cpu into lowest power state possible when system is idle.
- Enabling drivers to drop into low power states when inactive.
- Frequency and voltage scaling to reduce power consumption at runtime.

What's next

- Migrate more drivers to manage their own power states following USB as the example
- Integrate voltage scaling for embedded platforms into cpufreq somewhere
- More userspace development: OHM, policy managers, HAL.
- **Linux powered devices have the longest battery life!**

Matthew Locke
Mlocke@embeddedalley.com
408-386-1482

Embedded Alley